A modular bio-inspired architecture for movement generation for the infant-like robot *iCub*

Sarah Degallier, Ludovic Righetti, Lorenzo Natale, Francesco Nori, Giorgio Metta and Auke Ijspeert

***Abstract*— Movement generation in humans appears to be processed through a three-layered architecture, where each layer corresponds to a different level of abstraction in the representation of the movement. In this article, we will present an architecture reflecting this three-layered organization and based on a modular approach to human movement generation. We will show that our architecture is well suited for the online generation and modulation of motor behaviors, but also for switching between motor behaviors. This will be illustrated respectively through an *interactive drumming* task and through switching between *reaching* and *crawling*. This latter task has been tested with the ODE-based simulator Webots™ while drumming has been implemented on the real robot *iCub*.**

I. Introduction

In the framework of the European project RobotCub [1], which aims at developing an infant-like robot, *iCub*, with the motor and cognitive abilities of a 2 years-old child, we are currently developing a functional model of the human motor system, that is an architecture reflecting the different processes involved in low-level movement generation. Our motor architecture will be integrated within a larger cognitive architecture developed by the RobotCub consortium [2].

To build our model of the human motor system, we define a three-layered architecture whose layers are referred to as the planner, the manager and the generator. Functionally, the planner (i.e the motor cortex in humans) builds the mental representation of the task. Indeed, according to Jeannerod, when choosing to perform a given action, representations that account for behaviors "*must carry internal models of how the external world is, how it will be modified by the action of the organism, and how the organism will be modified by that action*" ([3],p.3). The manager (the brain stem, the basal ganglia and the cerebellum in humans) is involved in the selection, timing and coordination of the appropriate behaviors (motor programs, see for instance [4], [5]). Finally, the generator (the spinal cord) generates trajectories through central pattern generators (CPGs), that we see as networks of neurons involved in the production of movement primitives (for a review on CPGs, see [6] or [7]).

Since our particular interest is on movement generation, we will not focus on the high cognitive abilities needed to define and choose the action; in terms of the architecture, we do not focus on the implementation of the planner. These issues are addressed by other partners in the framework of the RobotCub project[1].

In order to develop an efficient model reflecting these principles, we make the assumption that movement generation is highly modular, both in terms of motor primitives (i.e. units of movement) and in terms of motor programs (i.e. behaviors), as will be discussed more in details in Section II. Indeed, modularity has proven to be a successful approach for generating fast, complex movements (see [8], [9], [10], [11]).

We assume the existence of two basic types of motor primitives, i.e. discrete (aperiodic and finite) and rhythmic (periodic) movements[2] (as done previously by [8], [13], [9]). We model these motor primitives as solutions of a dynamical system with a globally attractive fixed point and an oscillator, respectively. Such an approach allows us to use the stability properties of dynamical systems to ensure a robust control of the movements. We use a system similar to the one that we had previously developed [14], [15] which allows the generation of discrete (i.e. short-term) and rhythmic movements and the combination of both (i.e. oscillations around time-varying offsets).

In this article, we present our current implementation of this functional architecture (Section II) and the future improvements that we are planning (Section VI). The current implementation allows for an easy and fast online modulation of trajectories as well as the possibility of easily switching between behaviors according to sensory information; this will be illustrated through two applications, namely *interactive drumming* (Section IV), i.e. a user can specify on the fly the score that the robot has to play, and the switching between *crawling*, *reaching* on the fours and *reaching while crawling* (Section V). Interactive drumming has been tested on the real robot while switching between behaviors has been tested using Webots™ [16], a physics based simulator . The infant-like robot *iCub* is briefly presented in Section III.

II. Presentation of the architecture

We present here the current implementation of the architecture, which is depicted in Fig. 1. Note that since it is an ongoing work, many improvements are planned and will be discussed in Section VI.

This work was supported by the European Commission's Cognition Unit, project no. IST-2004-004370: RobotCub and by the Swiss National Science Foundation

S. Degallier, L. Righetti and A. Ijspeert are with the School of Computer and Communication Science, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, sarah.degallier@epfl.ch

L. Natale, F. Nori and G. Metta are with the Italian Institute of Technology, Genova, Italy

[1]See http://www.robotcub.org/misc/review3/index.html for a complete list of publications

[2]Note that Schaal et al. [12] have shown that rhythmic movements are not a particular case of discrete movements by using fMRI techniques; some brain areas involved in discrete task are not active during rhythmic movements.

Overall, we want to design an architecture that allows for the generation of complex, adaptive behaviors and for switching between these behaviors. This requires the ability to pertinently integrate sensory feedback and high level commands, but also the ability to deal with constraints.

In order to do so, we propose here a three layer architecture where each layer is an independent process that communicates with the others (see Fig. 1). Such a structure allows for the distribution of the tasks relatively to their cognitive level, but also to their computational need. Indeed, the role of the generator consists in our case only in integrating the dynamical systems, a task which requires low computational needs and can be implemented on the DSP controllers of the robot with fast feedback loops. Such an approach ensures that the generation of the trajectories is not perturbed by highly demanding processes such as optimization and planning which are solved at higher levels (i.e. the planner in our case). The manager ensures the coherence of the movements (both in terms of spatial and time constraints) and the communication between the high and the low level processes.

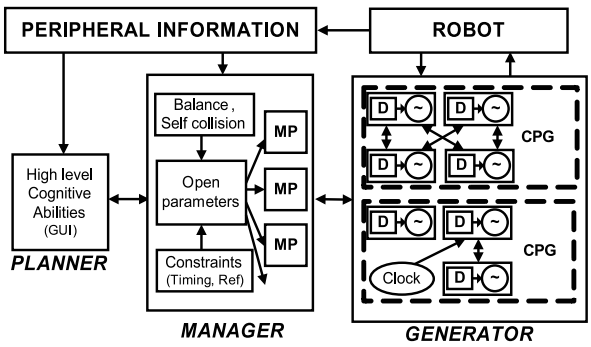


Fig. 1. Schematic of the functional organization of the architecture. At the manager level, the MPs are launched by the manager according to the contextual information and the commands sent by the planner. At the generator level, the unit generators, composed of a discrete system (D) and an oscillator, are coupled in network to form CPGs. Two possible types of coupling are represented. As said in the text, a clock can be added if an absolute timing is needed.

We now present in more details our implementation of the three layers. As stated before, we focus mainly on the low-level motor generation, i.e. on the manager and the generator. Feedback issues will be discussed in Section VI.

A. Generator

The generator, which is responsible for the generation of the trajectories, is built on the concept of central pattern generators (CPGs), that we take in the sense of a network of unit generators (UGs) of basic movements called motor primitives.

Unit generators are modeled in our architecture by dynamical systems; two types of primitives are defined, namely discrete and rhythmic, that correspond respectively to the solution of a globally attractive fixed point system and of a limit cycle system. The two main advantages of using such dynamical systems is that (i) the trajectories are generated online by integration, and thus their parameters can be modified smoothly on the fly, and (ii) the solutions obtained are robust to perturbations, and thus feedback can be easily included.

All trajectories (for each joint) are generated through a unique set of differential equations, which is designed to produce complex movements modeled as a periodic movement around a time-varying offset. More precisely, complex movements are generated through the superimposition and sequencing of simpler motor primitives generated by rhythmic and discrete unit generators. The discrete primitive is injected in the rhythmic primitive as an offset, although other combinations of them could be considered such as a sequencing or a simple addition of their output.

Thanks to the use of limit cycle systems, the different unit generators of each joints can be coupled in a network to obtain a more complex, synchronized behaviors. Such networks, that we call central pattern generators (CPG), are well suited to ensure fixed time relations between the different rhythmic outputs[3], a feature which is particularly convenient for generating different gaits for locomotion, for instance (see [18]). A reference limit cycle system can be added in the system to serve as a clock (as we did in drumming for instance).

Discrete UG.The discrete UG, which is inspired from the VITE model [19], is modeled by the following system of equations

$$\dot{h}_i = d(p - h_i) \quad (1)$$

$$\dot{y}_i = h_i^4 v_i \quad (2)$$

$$\dot{v}_i = p^4 \frac{-b^2}{4} (y_i - g_i) - b\, v_i. \quad (3)$$

The system is critically damped so that the output y_i of Eqs 2 and 3 converges asymptotically and monotically to a goal g_i with a speed of convergence controlled by b, whereas the speed v_i converges to zero. p and d are chosen so to ensure a bell-shaped velocity profile; h_i converges to p and is reset to zero at the end of each movement.

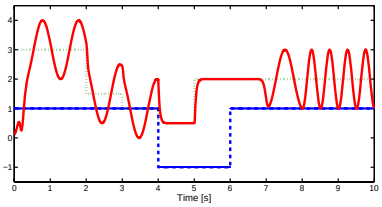


Fig. 2. Using simple variations of m_i (dash line), g_i (dotted line) and ω_i (not represented on the figure), a periodic trajectory around a time-varying offset can be generated. Setting m_i to a negative value turns off the oscillatory behavior thanks to the Hopf bifurcation.

Rhythmic UG. The rhythmic UG is modeled as a modi-

[3]The interested readers are referred to the work of Golubitsky for the construction of networks of coupled oscillator, see for instance [17].

fied Hopf oscillator:

$$\dot{x}_i = a\left(m_i - r_i^2\right)(x_i - y_i) - \omega_i z_i \quad (4)$$

$$\dot{z}_i = a\left(m_i - r_i^2\right) z_i + \omega_i (x_i - y_i) + \sum k_{ij} z_j + u_i \quad (5)$$

$$\omega_i = \frac{\omega_{down}}{\mathrm{e}^{-f z_i} + 1} + \frac{\omega_{up}}{\mathrm{e}^{f z_i} + 1} \quad (6)$$

where $r_i = \sqrt{(x_i - y_i)^2 + z_i^2}$. When $m_i > 0$, Eqs. 4 and 5 describe a Hopf oscillator whose solution x_i is a periodic signal of amplitude $\sqrt{m_i}$ and frequency ω_i with an offset given by g_i. A Hopf bifurcation occurs when $m_i < 0$ leading to a system with a globally attractive fixed point at (g_i,0). The term $\sum k_{ij} z_j$ controls the couplings with the other rhythmic UGs j; the k_{ij}'s denote the gain of the coupling between the rhythmic UGs i and j. The expression used for ω_i allows for an independent control of the speed of the ascending and descending phases of the periodic signal, which is useful for adjusting the swing and stance duration in crawling for instance [18]. Finally the term u_i is a control term generated by feedback information [20].

Qualitatively, by simply modifying on the fly the parameters g_i and m_i, the system can thus switch between purely discrete movements ($m_i < 0, g_i \neq cst$), purely rhythmic movements ($m_i > 0, g_i =$ cst), and combinations of both ($m_i > 0, g_i \neq cst$) as illustrated on Fig. 2 (see [14] for more details). More elaborate movements can be achieved by sending time-varying parameters ($\vec{m}(t), \vec{g}(t), \vec{\omega}(t)$) to the system and by integrating feedback signals to the generator, as will be described in Section V.

B. Manager

The manager is built upon the concept of motor program, which is defined as "*a set of muscle commands which are structured before a movement begins and which can be sent to the muscle with the correct timing so that the entire sequence is carried out in the absence of peripheral feedback*" by Marsden et al. [21]. This concept is a nice way of explaining the rapidity with which we react to stimuli and the stereotypy present in human movements. Moreover, the notion of generalized motor program (MP), that is motor programs with open parameters, allows the generation of movements adapted to the environment (see [5] for instance).

Functionally speaking, the manager is mainly responsible for sending the right parameters (in joint space) to the generator, at the right timing. We define a (generalized) motor program (MP) as a sequence of parameters sent to the generator to produce the desired trajectories, that is in our case $\vec{g}(t)$, $\vec{m}(t)$, $\vec{\omega}(t)$ and the couplings between the oscillators (i.e. the topology of the network). Some of the parameters are fixed (the coupling between the limbs for crawling for instance), others are open and need to be defined relatively to the environment and the task (the desired angles in reaching). An inverse kinematics is also needed to transform task space goals into target joint angles.

Every time a MP is launched by the manager, the first command sent corresponds to a predefined initial position. The parameters are then sent at regular time intervals to the generator. At the end of the sequence, a command corresponding to a final target position is sent. This makes the switching between tasks easier, as will be illustrated with crawling and reaching. A MP can be elicited, and interrupted, either by the planner (voluntary movements) or by the contextual sensory information (automatisms).

C. Planner

The planner is for now a simple GUI that allows the user to specify the task the robot has to perform in terms of cartesian space. However, higher cortical abilities can be implemented to define the task to be performed. The output that should be given to the manager is the target goal (or the target trajectory) in cartesian space as well as the parameters of the rhythmic movement.

III. Presentation of *iCub*

The *iCub* is the humanoid robot developed as part of the RobotCub project [1]. It has been designed to mimic the size of a three and a half year old child (approximately 1m tall). It has 53 degrees of freedom. A good part of them are allocated to the upper torso, especially to the hands (18 in total) to allow manipulation of objects. The *iCub* is strong enough to crawl on all fours and sit to free the hands for manipulating objects.

A. Hardware specifications

The *iCub* is based on electric motors for actuation. The major joints are actuated by brushless DC motors coupled with frameless Harmonic Drive gears. This guarantees torques up to 40Nm at the shoulders, spine and hips. The head and hands are actuated by smaller brushed-DC motors.

The robot is equipped with cameras, microphones, gyroscopes & linear accelerometers, force/torque sensors, position and temperature sensors. A fully sensorized skin and fingertips are under development.

The electronics of the *iCub* have been developed specifically to fit within the limited space available. Each controller card runs a 1KHz (position or velocity) control loop on a dedicated DSP. All cards are connected to a main relay CPU via four CAN bus lines. These lines end into a multi-purpose I/O card which communicates to the relay CPU (PC104) located in the robot head. More demanding computation can be performed on a PC cluster connected to the PC104 via a gigabit ethernet.

Additional electronics have been designed to sample and digitize the *iCub* sensors. Also in this case, everything converges to the PC104 by means of various additional connections (e.g. serial, firewire, etc.).

B. Software architecture

The *iCub* software architecture uses YARP, an open source library written to support software development and integration in robotics [22]. The core of YARP is an inter-process communication layer which allows processes on different machines to exchange data across an Ethernet network. Communication in YARP is transport independent; details

about the underlying network and protocol are hidden to the user. Similarly, YARP offers device driver wrappers, which help separating user-level code from vendor-dependent code related to sensors and actuators. Overall this contributes to achieve loose coupling between algorithms and hardware, and, in turn, favors modularity. In short, communication in YARP takes place through connections, called ports. Ports are named entities which move data from one process to another (or several others).

iCub capabilities are implemented as a set of modules, interconnected through YARP ports. Each module is an executable which implements a given functionality, and creates a set of ports to receive and send data. Some modules provide access to the hardware. For example the iCubInterface module exports a set of ports to give access to the motors and broadcast the encoder feedback from all joints. Other modules in the architecture control the robot by sending messages to these ports. Commands can be specified as joint space position or velocity.

IV. Application to interactive drumming

Interactive drumming is an interesting task combining discrete and rhythmic movements and requiring all four, precise timing, coordination between limbs and also the online modulation of the trajectories subject to constraints. As it does not require high level cognitive abilities or balance issues, it is well suited for a first test of the architecture on the real robot.

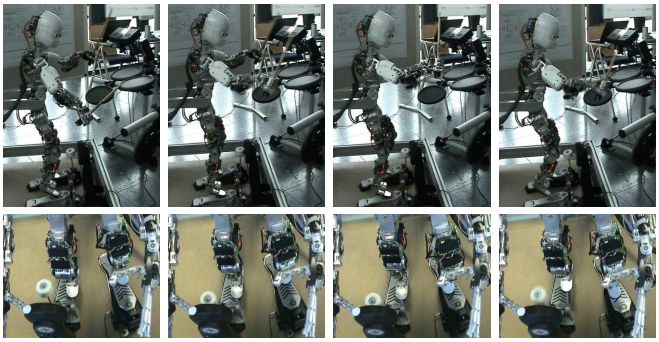


Fig. 3. Snapshots of the iCub drumming at the conference CogSys 2008. Top: Side view of the complete robot. Bottom: Downward view of the legs hitting the pedals.

Drumming has been implemented on the real iCub and presented as a demonstration at the CogSys 2008 conference in April (see Fig. 3 for some pictures of the setting). The robot is fixed to a metallic structure by the hips and plays on an electronic drum set. The four limbs together with the head are controlled. The sticks are grasped by the hands which remain fixed afterwards. We control actively four joints for each limb and one for the head.

The planner is simply implemented as a GUI that allows a user to define the score that the robot has to play; more precisely, the user can specify either a predefined score or the drum that has to be beaten next by each of the limbs. There is a "Hold" position corresponding to the mode where the limb does not beat anything. Moreover, the coordination between the limbs (i.e. their phase relation) and the frequency of the beating can be modified on the fly.

At the manager level, there is a unique motor program for each limb (MP) whose parameters are controlled through the GUI. The manager is then responsible for translating the commands sent by the planner in terms of joint space parameters for the generator. In the current application, the target discrete postures for hitting each drum are predefined; the integration of visual localization of the drums is planned as a future work. These parameters are sent at a specific timing corresponding to the beat (i.e. the tempo) of the score, this beat being given by the clock at the generator level.

Concerning the generator, each dof is controlled by the discrete and rhythmic pattern generators that we have presented in Section II. The couplings between the dofs, which are specified through a configuration file, are the following:

- for the legs, both hips are unilaterally coupled to the clock and bilaterally coupled to the knees;
- for the arms, both shoulders are unilaterally coupled to the clock and bilaterally coupled to the elbows.

The bilateral couplings between limbs allows for a precise synchronization between them. After a Hopf bifurcation, one can observe a phase resetting of the oscillators; the clock can be seen as a metronome that ensures that the limbs stay in synchronization with the absolute tempo despite those phase resettings.

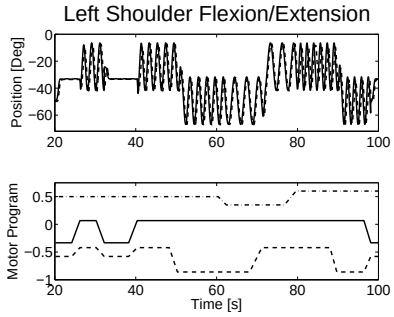
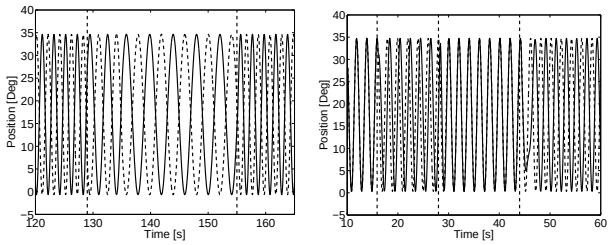


Fig. 4. Up: Trajectories generated by the generator for one arm obtained with iCub when drumming. Plain lines are desired trajectories and dotted lines are the actual trajectories. Down: Corresponding parameters sent by the manager to the generator.

Results. The implementation of the real *iCub* has successfully shown that the architecture was well-suited to allow for the online modulation of trajectories subject to time constraints (Fig. 4) as well as for the generation of synchronized movements between the limbs (Fig. 5). See [23] for a movie of the robot drumming.

On Fig. 4, it can be seen that the parameters are modulated in real time and that those modulations end up in a smooth adaptation of the generated trajectories. Moreover, the modifications occurs at specific times corresponding to the end of a beat thanks to the manager that deals with time constraints.

On Fig. 5, trajectories from the two legs are shown to illustrate coordination between limbs. It can be seen that



(a) Modulation of the frequency (b) Modulation of the coordination

Fig. 5. (a) The left leg (plain line) and the right leg (dash line) are in anti-phase. This phase shift remains constant even when frequency of the system is modified (at 130s and 155s, vertical dash line). Moreover, the convergence to the new frequency is less than on cycle. (b) The left and the right legs (resp. plain and dash lines) are in phase at the beginning of the movement. Then at time 15s, 28s, and 44s (vertical dash lines) the phase shift of the right leg relatively to the clock is modified. The trajectory converges in less than a cycle to the desired one.

the two legs stay synchronized even when the frequency is changed (Fig. 5(a)). Moreover, when the coordination of the legs is changed, the transition is fast and the trajectories remain smooth (Fig. 5(b)).

To enhance the performance, the addition of several types of feedback is planned in the future, as will be discussed in Section VI.

V. Application to Crawling and Reaching

In this application, we want to test the ability of the architecture to switch and combine behaviors. Contrarily to the drumming task, here behaviors are triggered by sensory information, i.e. no planner is involved. More precisely, we define three tasks (motor programs): *reaching*, *crawling* and *reaching while crawling*; each of this task is triggered by color marks on the ground, i.e. a red mark on the ground launches *reaching*, a blue mark *reaching while crawling* and no mark *crawling*. No visual processing is considered here; the position and color of the mark are directly provided to the robot. The robot crawls in an environment where it has to switch between those three behaviors according to marks arbitrarily placed on the ground. Combinations of *crawling* and *reaching* have been tested in simulation using the ODE-based software Webots™.

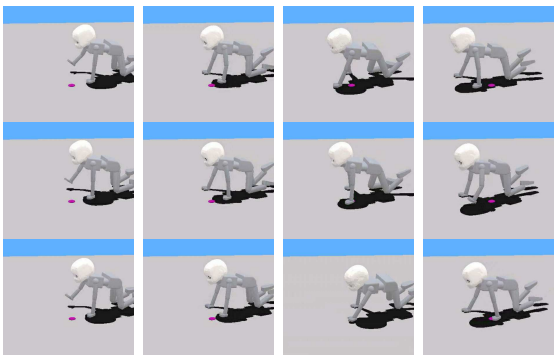


Fig. 6. Snapshots of the three behaviors with feedback. Upper line: Only crawling; middle line: Crawling while reaching; bottom line: iCub crawls, stops and then reaches the mark.

Each behavior is simply triggered through the specification of the amplitudes $\vec{m}$ and the offsets $\vec{g}$ by the manager ($\vec{\omega}$ is fixed). More precisely, the three motor programs are the following:

- ***Crawling*** ($\vec{m} > \vec{0}, \vec{g} = \vec{0}$). Analysis of crawling infants has shown that most infants crawl on hands and knees, using a walking trot gait [20]. The couplings are thus set so to obtain a trot gait and g is set to a fixed value (0 here) so to obtain a purely rhythmic movement.
- ***Reaching*** ($\vec{m} < \vec{0}, \vec{g} \neq \vec{0}$). Once a mark is close enough to be reachable, $\vec{m}$ is turned to a negative value to stop *crawling*. A target position $\vec{g}$ suitable to reach the mark, calculated using an inverse kinematic, is then sent to the generator.
- ***Reaching while crawling*** ($\vec{m} > \vec{0}, \vec{g} = \vec{g}(t)$). When a mark is reachable and when the arm is in an appropriate state (i.e. is in the swing phase), the desired position g is sent to the generator by the manager; the actual position of the system is then compared to the desired position so to reach the correct position through a modification of the offset (see [15] for details). $\vec{m}$ is kept to a positive value so that the resulting movement is rhythmic with a time varying offset.

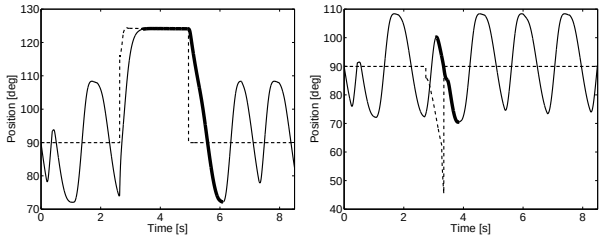
Feedback integration. A phase dependent sensory feedback is also included in the rhythmic PG to make the crawling locomotion more robust and adaptive to the environment. Information from the load sensors located on the hands and knees of the robot is used to modulate the onset of the swing and stance phases, as mammals do [24]. Depending on the values of the sensors and of the phase of the limb, the term u_i of Eq 5 is defined as

$$u_i = \begin{cases} -\text{sign}(y_i)F & \text{fast transitions} \\ -\omega x_i - \sum k_{ij} y_j & \text{stop transition} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

where F controls the speed of the transition. The feedback term modifies the phase plan of the oscillator according to the following rule: the transition from stance to swing phases is delayed as long as the other limbs cannot support the body weight (using the feedback term for fast transition) and is triggered sooner when the limb leaves unexpectedly the ground (using the feedback term to stop the transition). An analogous policy is used for the swing to stance transition. More details can be found in [25].

Results. Results obtained in simulation have shown that the architecture allows for smooth transitions between motor behaviors; in both *reaching while crawling* and *reaching*, the trajectory smoothly resume to crawling after the mark has been reached. Fig. 6 shows some snapshots of the three tasks; for the corresponding movies, see [26].

Note that the implementation that we are using here is slightly different from the one that we presented in [15] as the discrete system is embedded in the rhythmic one rather than added to it and as we have introduced a feedback term here to deal with the perturbation introduced in the different phase of locomotion by the modification of the offset of the reaching limb. Thanks to this feedback term, the resulting trajectory is smoother (as can be seen on the movie at [26])



(a) Reaching (b) Reaching while Crawling

Fig. 7. Trajectories the shoulder felxion/extension joint (left arm, plain line) and the corresponding parameter g (dotted). Bold line means that the mark is reached. (a) The robot switches from crawling to reaching when the mark is reachable and then start crawling again.(b) The robot reaches the mark while crawling through a modification of the offset of the oscillation.

and we were able to remove some constraints that were added to prevent self collision.

VI. Discussion

The architecture that we have presented has been proven to be successful for both the specification on the fly of a behavior as well as the switching between behaviors accordingly to sensory information. Moreover, the architecture is suitable for the online generation of complex trajectories and can serve as the low level basis for any type of motor behaviors just by modifying the planner layer.

Different improvements of the architecture are planned in the future, among which the integration of several feedback signals (both at the generator and at the manager level) and the integration of constraints such as balance and self collision avoidance in the manager, and joint limits in the generator.

At the generator level, feedback from touch sensors has already been implemented and tested in a simulation environment, as discussed in Section V. We are currently adding a force feedback to detect collisions; this feedback is modeled as a repulsor which is activated when a sudden increase in the force sensor is detected. Such a feedback results in slowing down rapidly the movement to maintain it in its current position. Note that as soon as the perturbation disappears, the movement resumes to its trajectory without time delay. Eventually, force feedback will also be tested on the *iCub* robot. At the manager level, we are going to implement a visual feedback to allow target and obstacle detection.

VII. Conclusion

We have presented here a three-layer architecture suitable for the generation of various motor tasks, as interactive drumming, reaching and crawling. It has been shown that it allows for the online specification of a given motor task as well as for switching between motor tasks. Moreover, the distributed nature of the architecture makes it well suited for its integration on real robots, as shown with the *iCub*.

References

[1] N.G. Tsagarakis, G. Metta, G. Sandini, D. Vernon, R. Beira, F. Becchi, L. Righetti, J. Santos-Victor, A.J. Ijspeert, M.C. Carrozza, and D.G. Caldwell. iCub - The Design and Realization of an Open Humanoid Platform for Cognitive and Neuroscience Research. *Journal of Advanced Robotics, Special Issue on Robotic platforms for Research in Neuroscience*, 21(10):1151–1175, October 2007.

[2] Giulio Sandini, Giorgio Metta, and David Vernon. The *cub* cognitive humanoid robot: An open-system research platform for enactive cognition. In *50 Years of Artificial Intelligence*, pages 358–369, 2006.

[3] M. Jeannerod. *The Cognitive Neuroscience of Action*. Oxford: Blackwell, 1997.

[4] E. R. Kandel, J.H. Schwartz, and T. M. Jessell. *Principles of Neural Science*. Mc Graw Hill, 2000.

[5] R.A. Schmidt and T.D. Lee. *Motor control and learning: A behavioral emphasis*. Human Kinetics, Champaign, IL, USA, 2005.

[6] S. Grillner. Biological pattern generation: The cellular and computational logic of networks in motion. *Neuron*, 52(5):751–766, December 2006.

[7] A. J. Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks (to appear)*, 2007.

[8] S. Schaal, S. Kotosaka, and D. Sternad. Nonlinear dynamical systems as movement primitives. In *International Conference on Humanoid Robotics (Humanoids00)*, pages 117–124. Springer, 2000.

[9] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In S. Thrun S. Becker and K. Obermayer, editors, *Neural Information Processing Systems 15 (NIPS2002)*, pages 1547–1554, 2003.

[10] M. Kawato. Learning internal models of the motor apparatus. In SP Wise JR Bloedel, TJ Ebner, editor, *The Acquistion of Motor Behavior in Vertebrates*, pages 409–430. Cambridge MA: MIT Press, 1996.

[11] J. Tani, Y. Ito, and Y. Sugita. Self-organization of distributedly represented multiple behavior schemata in a mirror system: reviews of robot experiments using rnnpb. *Neural Networks*, 17:1273–1289, 2004.

[12] S. Schaal, D. Sternad, R. Osu, and M. Kawato. Rhythmic arm movement is not discrete. *Nat. Neuroscience*, 7(10):1136–1143, 2004.

[13] D. Sternad, W.J. Dean, and S. Schaal. Interaction of rhythmic and discrete pattern generators in single joint movements. 19:627–665, 2000.

[14] S. Degallier, C. P. Santos, L. Righetti, and A. Ijspeert. Movement generation using dynamical systems: a humanoid robot performing a drumming task. In *IEEE-RAS Inter. Conf. on Humanoid Robots*, pages 512–517, 2006.

[15] S. Degallier, L. Righetti, and A. Ijspeert. Hand placement during quadruped locomotion in a humanoid robot: A dynamical system approach. In *IEEE-RAS International Conference on Intelligent Robots and Systems (IROS07).*, 2007.

[16] O. Michel. Webots tm: Professional mobile robot simulation. *International Journal of Advanced Robotic System*, 1:39–42, 2004.

[17] M. Golubitsky, I. Stewart, and A. Torok. Patterns of synchrony in coupled cell networks with multiple arrows. *SIAM J. Appl. Dynam. Sys.*, 4(1):78–100, 2005.

[18] L. Righetti and A.J. Ijspeert. Design methodologies for central pattern generators: an application to crawling humanoids. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[19] D. Bullock and S. Grossberg. The vite model: a neural command circuit for generating arm and articulator trajectories. pages 206–305, 1988.

[20] L. Righetti, A. Nylén, K. Rosander, and A.J. Ijspeert. Kinematics of crawling in infants. In *preparation*.

[21] C.D. Marsden, P.A. Merton, and H. Morton. The use of peripheral feedback in the control of movements. *Trends Nerosci.*, 7:253–258, 1984.

[22] Paul Fitzpatrick, Giorgio Metta, and Lorenzo Natale. Towards long-lived robot genes. *Robot. Auton. Syst.*, 56(1):29–45, 2008.

[23] Movie of Drumming. `http://birg2.epfl.ch/users/degallier/movies_BioRob/icubdrum.mpg`.

[24] S. Frigon and S. Rossignol. Experiments and models of sensorimotor interactions during locomotion. *Biological Cybernetics*, 95(6):607–627, 2006.

[25] L. Righetti and A.J. Ijspeert. Pattern generators with sensory feedback for the control of quadruped locomotion. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation ICRA*. Accepted.

[26] Movie of Crawling and Reaching. `http://birg2.epfl.ch/users/degallier/movies_BioRob/crawl.avi`.